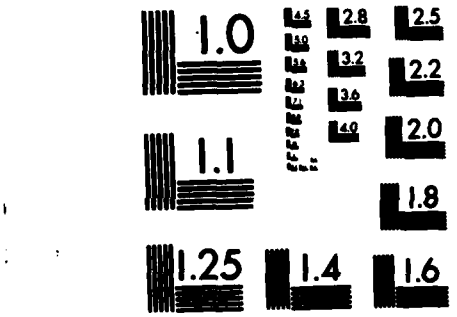
AD-A124 491 MINIMIZING ACCESS POINTER INTO TREES AND ARRAYS(U) 1/1
ILLINOIS UNIV AT URBANA APPLIED COMPUTATION THEORY
GROUP M C LOUI JUN 81 ACT-27 N00014-79-C-0424

UNCLASSIFIED F/G 9/4 • NL

END
DATE FILMED
8 31
DTIC



UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A124491	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) MINIMIZING ACCESS POINTERS INTO TREES AND ARRAYS	5. TYPE OF REPORT & PERIOD COVERED Technical Report
	6. PERFORMING ORG. REPORT NUMBER R-910; ACT 27; UILU-ENG 81-2241
7. AUTHOR(s) Michael C. Loui	8. CONTRACT OR GRANT NUMBER(s) N00014-79-C-0424; MCS-8010707
9. PERFORMING ORGANIZATION NAME AND ADDRESS Coordinated Science Laboratory University of Illinois 1101 W. Springfield Avenue Urbana, IL 61801	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Joint Services Electronics Program; National Science Foundation (MIT, Cambridge, MA).	12. REPORT DATE June 1981
	13. NUMBER OF PAGES 29
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report) UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Tree, array, pointer, data structure, tree machine, multidimensional Turing machine, simulation.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Multihead tree machines and multihead multidimensional machines are used to develop new methods for minimizing access pointers into trees and arrays. Every multihead tree machine of time complexity $t(n)$ can be simulated on-line by a tree machine with only two access heads in time $O(t(n)\log t(n)/\log\log t(n))$. Every multihead e-dimensional machine of time complexity $t(n)$ can be simulated on-line $O(t(n)^{1+1/d-1/de}\log t(n))$. The simulation for trees is optimal.

DD FORM 1 JAN 73 1473

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

MINIMIZING ACCESS POINTERS INTO TREES AND ARRAYS

Michael C. Loui

Coordinated Science Laboratory
University of Illinois
Urbana, Illinois 61801

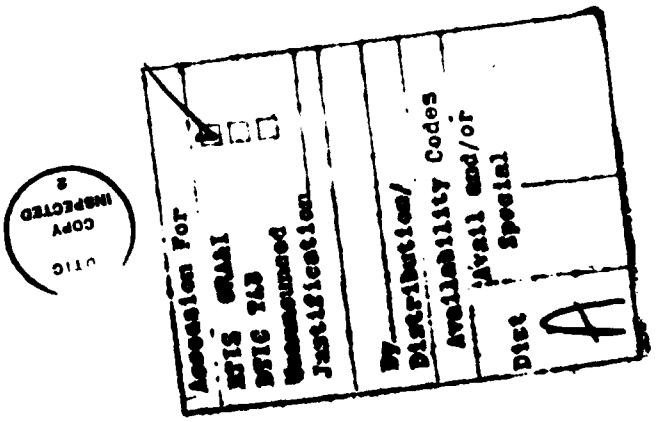
June 1981

ABSTRACT

Multihead tree machines and multihead multidimensional machines are used to develop new methods for minimizing access pointers into trees and arrays. Every multihead tree machine of time complexity t(n) can be simulated on-line by a tree machine with only two access heads in time $O(t(n)\log t(n)/\log\log t(n))$. Every multihead e-dimensional machine of time complexity t(n) can be simulated on-line by a d-dimensional machine with two access heads in time $O(t(n)^{1+1/d-1/de}\log t(n))$. The simulation for trees is optimal.

Key Words: Tree, array, pointer, data structure, tree machine, multidimensional Turing machine, simulation.

Supported by the Joint Services Electronics Program (U.S. Army, U.S. Navy, and U.S. Air Force) under Contract N00014-79-C-0424. Part of this work was performed at the Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts with the support of the National Science Foundation under Grant MCS-8010707.



1. INTRODUCTION

A data structure with local access premits an access pointer into the structure to change its value from one location only to an adjacent location in the structure. For example, trees are typically implemented with link fields in each record to the parent and the children; after information in a record is retrieved, the next record accessed is one of these adjacent records. Even arrays may have local access if implemented by orthogonal lists [K1] rather than random access to entries by address.

Multihead Turing machines model storage and retrieval operations in data structures with local access: tree machines for trees, multidimensional machines for arrays. The access heads of the machines correspond to the access pointers into the data structures. At each step, the machine can shift each access head only to an adjacent cell in its worktape storage. Though possibly unrealistic, the simple input/output and control conventions are not essential for our results, which depend only on the storage structure and the local access method.

Paul, Seiferas, and Simon [PSS] established nonlinear lower bounds on the time required to simulate tree machines on-line by tree machines with fewer access heads and to simulate multidimensional machines on-line by multidimensional machines with fewer access heads. In essence, increasing the number of access pointers into a tree or an array can save time.

We present efficient simulations of multihead machines by machines with fewer access heads.

Theorem 1. Every multihead tree machine of time complexity $t(n)$ can be simulated on-line by a tree machine with two access heads on separate worktapes in time $O(t(n)\log t(n)/\log\log t(n))$.

Theorem 2. For all $d \geq 2$ and $e \geq 2$, every multihead e-dimensional machine of time complexity $t(n)$ can be simulated on-line by a d-dimensional machine with two access heads on separate worktapes in time $O(t(n)^{1+1/d-1/de} \log t(n))$.

(Slightly stronger versions of Theorems 1 and 2 appear as Theorems 3 and 4 below.)

The simulation for tree machines is optimal (within a constant factor). Theorem 1 achieves the lower bound $\Omega(t(n)\log t(n)/\log\log t(n))$ of Paul, Seiferas, and Simon.

The simulation for multidimensional machines is not known to be optimal. The known lower bound on the time required for simulating e-dimensional machines by d-dimensional machines with just two access heads on-line is $\Omega(t(n)^{1+1/d-1/de-\delta})$ for every $\delta > (d-1)/de(e+1)$ and $d \geq 2$ [PSS]. Nevertheless, our simulation is faster than the simulation of Paul, Seiferas, and Simon, which for $e = d \geq 2$ runs in time $O(t(n)^{1+1/d-\epsilon})$, $\epsilon = 1/(d^2(d-1)+d)$. Furthermore, our simulation employs two data representations, converting from one to the other when necessary. This technique may be useful in other situations. For the case $d = 1$, Stoss [S] generalized a classic result of Hennie and Stearns [HS] and discovered a simulation of an e-dimensional machine by a one-dimensional machine with two access heads on separate worktapes in time $O(t(n)^{2-1/e} \log t(n))$.

2. TREE MACHINES

Tree machines generalize conventional multihead Turing machines with linear tapes. A tree worktape is a collection of cells organized into a complete infinite rooted binary tree. The root has two children, and all other cells each have two children and a parent. A tree machine has a finite-state control, a read-only linear input tape, a write-only linear output tape, and a finite number of tree worktapes, each of which has a finite number of access heads. At each step the machine reads the symbols in the cells on which the input and worktape heads are positioned, writes symbols from a finite alphabet on these worktape cells and possibly on the output tape too, and possibly shifts each head to an adjacent cell. Initially, all worktape calls hold blanks, and every worktape head is positioned on the root of its tape. The machine can detect when one of its heads is at the root and when two heads scan the same cell.

Let W be a tree worktape. To each cell of W assign in a natural way a string in $\{0,1\}^*$ called the location of the cell. Write W(b) for the cell of W at location b. Let λ denote the empty string. Then $W(\lambda)$ is the root of W. Write W[b,r] for the complete subtree of W of height r rooted at W(b). Cell W(a) is at distance r from cell W(b) if an access head requires exactly r steps to move from W(a) to W(b). For example, the leaves of W[b,r] are at distance r from W(b).

Let T be a tree machine. To establish Theorem 1, we devise an on-line simulation of T by a tree machine S with two heads on separate worktapes. Recall that in general, machine M' simulates machine M of time complexity

t(n) <u>on-line in time f(t(n))</u> if on every input word, if $s_1 \le s_2 \le \ldots \le s_k$ are the steps at which M reads or writes a symbol, then there are corresponding steps $s'_1 \le s'_2 \le \ldots \le s'_k$ at which M' reads or writes the same symbols, and every $s'_i \le f(s_i)$.

Consider a computation of N steps on T. For simplicity, we present a simulation in which N is available off-line. The simulation can be converted to an on-line simulation with time loss of only a constant factor [PSS]. (For N' = 1,2,4,8,..., machine S repeats the simulation from the beginning using N' for the length of the computation until N' is at least as large as N.) Without loss of generality, assume that every worktape head of T remains within distance O(log N) of the root of its tape. Section 2.3 justifies this assumption. Also, assume that the h access heads of T are on separate worktapes $W_1,\ldots,W_h$. Section 2.2 handles worktapes with multiple heads.

2.1 Simulation

Call one worktape of S the <u>primary</u> tape P, the other the <u>secondary</u> tape. Section 2.2 replaces the secondary tape by a linear tape. Divide the cells of P into 4h tracks, four for each worktape of T, numbered $C_1,\ldots,C_h$ (worktape contents), $H_1,\ldots,H_h$ (head positions), $C'_1,\ldots,C'_h$ (local contents), $H'_1,\ldots,H'_h$ (local head positions). In the initial configuration of S, the blank symbol in every cell is interpreted as a blank symbol on every track.

Set r = (log log N)/2. The N steps of the computation of T are simulated in N/r phases of r steps each. At the end of the simulation of each phase, track C_i of P holds the simulated contents of W_i; for every b, track C_i

of P(b) holds the same symbol as $W_i(b)$ would. Furthermore, at the end of each simulated phase, track H_i indicates a path from the root of P to $P(x_i)$ if the simulated head on W_i is positioned on $W_i(x_i)$.

To simulate one phase, S first determines the location a_i of the ancestor at distance r from the current location x_i of the simulated access head on W_i, setting $a_i = \lambda$ if $W_i(x_i)$ is within distance r of the root $W_i(\lambda)$. Next, S copies the contents of tracks C_i and H_i of $P[a_i,2r]$ to the corresponding tracks C_i' and H_i' of $P[\lambda,2r]$, using the secondary tape for intermediate storage. Observe that the access head on W_i remains in $W_i[a_i,2r]$ during this phase of r steps. Thus, when S completes these copying operations, tracks $C_1',\ldots,C_h'$ of $P[\lambda,2r]$ hold the contents of all cells that the simulated heads of T could access during the phase.

To simulate one step of T, machine S employs tracks C_i' and H_i' of $P[\lambda,2r]$ to determine for each i the contents of the cell read by the simulated head on W_i. Machine S then records the new contents of the cell on track C_i' and the new head position on track H_i'. (The management of track H_i' is entirely routine.) At the end of the phase, for each i, S uses the secondary tape to copy the contents of tracks C_i' and H_i' of $P[\lambda,2r]$ back onto tracks C_i and H_i of $P[a_i,2r]$.

Let us reckon the time consumed by S. At the beginning and end of each phase, S moves its head on the primary tape to each $P(a_i)$ (time $O(\log N)$ by hypothesis on T). It copies the contents of h trees of $O(4^r)$ cells each to the top of P and back (time $O(4^r)$). To simulate one step, S twice moves the head to a cell on $P[\lambda,2r]$ for each simulated head of T (time $O(r)$). Therefore, S runs in time

$$(N/r)[O(\log N)+O(4^r)+rO(r)] = O(N \log N/\log\log N).$$

2.2 Multihead Tapes and a Linear Secondary Worktape

We modify S to simulate T when T has multihead tapes. Clearly, we may assume that the h access heads of T are on the same worktape W. Divide the primary worktape of S into $3h+1$ tracks called C, $H_1,\ldots,H_h$, $C'_1,\ldots,C'_h$, $H'_1,\ldots,H'_h$. As before, at the end of the simulation of a phase, track C holds the contents of the simulated worktape W, and tracks $H_1,\ldots,H_h$ specify the access head positions. Tracks $C'_1,\ldots,C'_h$, $H'_1,\ldots,H'_h$ are used to simulate individual steps.

Machine S sets $r = (\log \log N)/2$ and simulates T in N/r phases of r steps each. To simulate a phase, S determines the location a_i of the ancestor at distance r from the current location of the i-th simulated access head of T. In time $O(4^r)$, S decides for each pair (i,j) whether $W[a_i,2r]$ meets $W[a_j,2r]$. If so, then because head j might read what head i writes during this phase, S copies track C of both $P[a_i,2r]$ and $P[a_j,2r]$ to the same C' track of $P[\lambda,2r]$. Since the time to simulate one step once the copying operations are completed is $O(hr) = O(r)$, this modified S still runs in time $O(N \log N/\log \log N)$.

(To determine whether $W[a_i,2r]$ meets $W[a_j,2r]$, S marks cells of $P[a_i,2r]$ and cells of $P[a_j,2r]$ and checks whether $P[a_i,2r]$ meets $P[a_j,2r]$. In its finite state control S constructs a graph with h vertices $\{1,\ldots,h\}$ and all edges (i,j) such that $W[a_i,2r]$ meets $W[a_j,2r]$. The connected components of this graph specify which $P[a_k,2r]$ should be copied to the same C' track of $P[\lambda,2r]$.)

Suppose T has head-to-head jumps: at each step head i can jump to the cell on which head j is located. In terms of data structures, pointer i can be assigned the current value of pointer j. To simulate this head-to-head jump, S records the new position of head i on track H_i' of P[λ,2hr] in time O(r). When completing the simulation of the phase, S adjusts the contents of track H_i in time O(log N).

We can replace the secondary tree worktape of S by a linear (one-dimensional) worktape without time loss. Initially, S writes a pattern of length $O(2^{2r})$ on the linear tape that describes a depth-first traversal of a complete binary tree of height 2r. This can be accomplished by using the primary tape as a linear tape to count to 2^{2r} in binary. To copy the contents of a tree P[b,2r] from the primary worktape, S copies the contents of the tree onto this pattern, which specifies the movement of the head on the primary worktape.

We summarize these alterations to S in a form that subsumes Theorem 1.

Theorem 3. Every multihead tree machine with head-to-head jumps of time complexity t(n) can be simulated on-line by a machine with a tree worktape and a linear worktape, each with one access head, in time O(t(n)log t(n)/log log t(n)).

2.3 Depth Reduction

We establish a depth reduction lemma for multihead tree machines. Paul and Reischuk [PR] obtained a similar result.

Lemma 1. Every multihead tree machine T of time complexity t(n) can be simulated on-line by a multihead tree machine S of time complexity O(t(n)) whose worktape heads remain within distance O(log t(n)) of the root.

Proof. As usual, assume that t is a power of 2 and is available off-line. Also, without loss of generality, assume that all h access heads of T are on one worktape W. The simulator S has several access heads, including three for each head of T, on one worktape V. Throughout the simulation these heads remain within distance O(log t) of the root of V.

Divide V into two tracks, numbered 0 and 1. In the initial configuration, a blank in a cell of V is interpreted as a blank on both tracks. During the simulation, track 0 holds the contents of cells of W, and track 1 holds control information.

Set $L = \log(ht)$. For all b, call $W[b,L]$ the top half of $W[b,2L+1]$ and the rest of $W[b,2L+1]$ its bottom half. Let $v_0, v_1, \ldots, v_{ht}$ be the locations of cells at distance L from the root, and set $V_i = V[v_i, 2L+1]$.

Track 0 of V_0 always represents the cells of $W[\lambda,2L+1]$ literally: for every cell $W(x)$ in $W[\lambda,2L+1]$, the symbols in $W(x)$ and track 0 of $V(v_0x)$ are the same. (v_0x is the concatenation of the binary strings v_0 and x.)

For $j > 0$, however, only the bottom half of V_j holds symbols written in W. When $a \neq \lambda$, the contents of $W[a,2L+1]$ correspond to V_i, V_j, where $i < j$, if

(i) there is a cell $V(u_j)$ in V_i at distance L from $V(v_i)$ such that v_j is written on track 1 of $V[u_j,L]$;

(ii) u_j is written on track 1 of $V[v_j,L]$; and

(iii) for all x, if $W(ax)$ is in the top half of $W[a,2L+1]$, then the symbol in $W(ax)$ is the same as the symbol on track 0 of $V(u_jx)$, whereas if $W(ax)$ is in the bottom half of $W[a,2L+1]$, then the symbol in $W(ax)$ is the same as the symbol on track 0 of $V(v_jx)$.

Throughout the simulation, S maintains the contents of W through these correspondences. Suppose either that $W[a,2L+1]$ corresponds to V_i,V_j or that $a = \lambda$ and $j = 0$. If $W(b) = W(ac)$ is at distance $L+1$ from $W(a)$ in $W[a,2L+1]$ and the bottom half of $W[b,2L+1]$ is nonblank, then track 1 of $V[v_jc,L]$ holds some v_k, and $W[b,2L+1]$ corresponds to V_j,V_k. See Figure 1. Observe that the relative position of v_jc in V_j is the same as the relative position of b in $W[a,2L+1]$. Conversely, if $V(v_jc)$ is at distance $L+1$ from $V(v_j)$ in V_j and track 1 of $V[v_jc,L]$ holds some v_k, then $W[ac,2L+1]$ corresponds to V_j,V_k.

For every access head H_m of T, machine S has access heads A_m (the top head) and B_m (the bottom head). Suppose $W[a,2L+1]$ corresponds to V_i,V_j. To simulate H_m reading cell $W(ax)$ using V_i,V_j, the top head A_m is at cell $V(u_jx)$ and the bottom head B_m at $V(v_jx)$. If $W(ax)$ is in the top half of $W[a,2L+1]$, then the top head A_m is in $V[u_j,L]$, and S retrieves the symbol in $W(ax)$ from track 0 of $V(u_jx)$. If $W(ax)$ is in the bottom half of $W[a,2L+1]$, then S uses the track 0 contents of $V(v_jx)$ read by the bottom head B_m. A third access head C_m is used by S as a unary counter to determine whether $W(ax)$ is in the top half or the bottom half of $W[a,2L+1]$.

In the step-by-step simulation of T, there are two situations in which A_m and B_m require <u>reorientation</u>.

(1) Let $W[a,2L+1]$ correspond to V_i,V_j, and let A_m,B_m use V_i,V_j. Suppose H_m attempts to visit a child of a leaf $W(az)$ of $W[a,2L+1]$. Let $W(b)$ be the ancestor of $W(az)$ at distance L from $W(az)$. Similarly, let $V(v_jc)$ be the ancestor of $V(v_jz)$ at distance L from $V(v_jz)$. Define d so that $cd = z$. See Figure 1. If track 1 of $V[v_jc,L]$ contains some v_k,

then W[b,2L+1] corresponds to V_j,V_k. In this case S sends A_m from $V(u_jz)$ to $V(v_jz)$ in V_j, sends B_m from $V(v_jz)$ to $V(v_kd)$ in V_k, and resumes the simulation with A_m,B_m using V_j,V_k. Otherwise, if track 1 of $V[v_jc,L]$ is blank, then S marks the first unmarked $V(v_k)$, writes v_k in track 1 of $V[v_jc,L]$ writes v_jc in track 1 of $V[v_k,L]$, and proceeds as above, for now W[b,2L+1] corresponds to V_j,V_k. (The contents of track 0 of V_k are completely blank.)

(2) Let W[b,2L+1] correspond to V_j,V_k, and let A_m,B_m use V_j,V_k. Suppose that H_m attempts to visit the parent of W(b). Let W(a) be the ancestor of W(b) at distance L+1 from W(b). Define c so that b = ac, and set $u_k = v_jc$. From track 1 of $V[v_j,L]$, S retrieves u_j, where $V(u_j)$ is in some V_i. We know that W[a,2L+1] corresponds to V_i,V_j. S sends head B_m from $V(v_k)$ to $V(u_k)$ and head A_m from $V(u_k)$ to $V(u_jc)$, and resumes the simulation with A_m,B_m using V_i,V_j.

Since for each access head H_m of T, machine S simulates at least $L = \Theta(\log t)$ steps between reorientations of A_m,B_m, and each reorientation takes time O(log t), we conclude that S simulates T in linear time. □

Corollary. Every multihead tree machine with head-to-head jumps of time complexity t(n) can be simulated on-line by a tree machine with head-to-head jumps of time complexity O(t(n)) whose worktape heads remain within distance O(log t(n)) of the root.

Proof. We modify the simulator S described in the proof of Lemma 1. Assume that the tree machine T has h heads on one worktape W. Both S and T permit head-to-head jumps. Set L = log(ht).

For each worktape head H_m of T, machine S has several heads on the same worktape V, including A'_m (ancestor top head) and B'_m (descendant bottom head) in addition to the top head A_m and the bottom head B_m described earlier. The heads A'_m and B'_m will assist in reorientations. We now specify when A'_m and B'_m are ready.

Suppose A_m, B_m use V_j, V_k to simulate H_m in W[b,2L+1], cell W(a) is the ancestor of W(b) at distance L+1 from W(b), and b = ac: the top head A_m is at $V(v_jcx)$, and the bottom head B_m is at $V(v_kx)$. See Figure 1. If A'_m is ready, then A'_m is at $V(u_jc)$, a descendant of $V(v_i)$, such that u_j is written on track 1 of $V[v_j,L]$. Note that if A_m, B_m used V_i, V_j to simulate H_m at W(b) in W[a,2L+1], then A_m would be at $V(u_jc)$.

Now suppose further that W(be) is at distance L+1 from W(b) and B_m is at $V(v_kex)$ in the bottom half of V_k. If B'_m is ready, then B'_m is at $V(v_\ell x)$ in V_ℓ, where W[be,2L+1] corresponds to V_k, V_ℓ. Note that if A_m, B_m used V_k, V_ℓ to simulate H_m in W[be,2L+1], then B_m would be at $V(v_\ell x)$.

The on-line simulation of T by S proceeds as before. Let W[b,2L+1] correspond to V_j, V_k, and let A_m, B_m use V_j, V_k. When H_m shifts to an adjacent cell in W[b,2L+1], heads A_m and B_m shift to the corresponding cells in V_j and V_k. Furthermore, if B'_m in V_ℓ is ready, and B_m remains in the bottom half of V_k, then B'_m also shifts to an adjacent cell in V_ℓ and remains ready.

To simulate a jump of H_m to H_p, all heads of S for H_m jump to the corresponding heads for H_p: head A_m jumps to A_p, head B_m to B_p, head A'_m to A'_p, and head B'_m to B'_p. If A'_p is ready, then S marks A'_m ready; if B'_p is ready, then S marks B'_m ready.

Suppose H_m shifts to the parent of W(b). If A'_m is ready at a descendant of $V(v_i)$, then head B_m jumps to A_m, head A_m jumps to A'_m, and the simulation continues with A_m, B_m using V_i, V_j; after this reorientation, heads A'_m and B'_m are no longer ready. If A'_m is not ready when H_m shifts to the parent of W(b), then a <u>widespread reorientation</u> is necessary to make A_m, B_m use V_i, V_j. Widespread reorientations are discussed below.

Suppose that W(be) is at distance L+1 from W(b) and H_m shifts to a child of a leaf W(bex) of W[b,2L+1]. If B'_m is ready in V_ℓ, then head A_m jumps to B_m, head B_m jumps to B'_m, and the simulation continues with A_m, B_m using V_k, V_ℓ; after this reorientation, heads A'_m and B'_m are no longer ready. Suppose B'_m is not ready when H_m shifts to a child of W(bex). If W[be,2L+1] corresponds to V_k, V_ℓ for some V_ℓ, then a widespread reorientation makes A_m, B_m use V_k, V_ℓ. Otherwise, the bottom half of W[be,2L+1] has not been visited previously, and an <u>initializing reorientation</u> is necessary to initialize a new V_ℓ to make A_m, B_m use V_k, V_ℓ.

As in the proof of Lemma 1, for an initializing reorientation S marks the first unmarked $V(v_\ell)$, writes v_ℓ in track 1 of $V[v_k e,L]$, and writes $v_k e$ in track 1 of $V[v_\ell,L]$. Then S sends head A'_m to $V(v_j ce)$, head A_m to $V(v_k ex)$ in V_k, and head B_m to $V(v_\ell x)$ in V_ℓ; it marks A'_m ready and B'_m not ready. Furthermore, for all other H_p, it makes ancestor top head A'_p ready and,

when possible, descendant bottom head B'_p ready. Observe that L cells in the top half of W[be,2L+1] must have been visited previously because the cells visited by heads of T always form a connected region of W. Consequently, during the simulation of T by S, there are only O(t/L) initializing reorientations, each of which takes time O(L). The total time for initializing reorientations is O(t).

When H_m induces a widespread reorientation, S uses information in track 1 to send A_m, B_m to the appropriate cells in subtrees V_i, V_j or V_k, V_ℓ. Moreover, for every p = 1,...,h, machine S makes ancestor top head A'_p ready and, when possible, descendant bottom head B'_p ready. It is crucial to realize that S can simulate at least L steps of T between widespread reorientations. For example, suppose that immediately after a widespread reorientation, A_p, B_p use V_j, V_k to simulate H_p in W[b,2L+1], and B'_p is not ready. Then either B_p is in the top half of V_k, or B_p is in $V[v_k e,L]$ in the bottom half of V_k for some W(be) at distance L+1 from W(b), and there is no V_ℓ such that W[be,2L+1] corresponds to V_k, V_ℓ. In the former case, H_p is at distance at least L+1 from a leaf of W[b,2L+1]. In the latter case, if H_p visited a leaf of W[b,2L+1], then it would induce an initializing reorientation, not a widespread reorientation; after the initializing reorientation, all ancestor top heads and as many descendant bottom heads as possible would be ready again.

Therefore, since S simulates at least L steps of T between widespread reorientations, there are only O(t/L) widespread reorientations, each of which takes time O(L). The total time for widespread reoreintations is O(t), and S simulates T in linear time. □

3. MULTIDIMENSIONAL MACHINES

The two tape simulation of a multidimensional machine devised by Paul, Seiferas, and Simon [PSS] uses pages of fixed size and a hierarchy of simulation procedures to maintain the locations of the pages. To develop a faster simulation, we employ a directory and a mechanism for handling pages of variable size [L]. In addition, our simulator uses both trie representations and literal representations for contents of the multihead machine's worktapes. During the simulation, it may convert the contents of a page from one representation into the other.

To each cell of a d-dimensional worktape assign in the usual way a d-tuple of integers called the coordinates of the cell. The origin is the cell whose coordinates are all zero. A d-dimensional Turing machine has a finite-state control, a read-only linear input tape, a write-only linear output tape, and a finite number of d-dimensional worktapes, each of which has a finite number of access heads. At each step the machine reads the symbols in the cells on which the input and worktape heads are positioned, writes symbols from a finite alphabet on these worktape cells and possibly on the output tape too, and possibly shifts each head to an adjacent cell. Initially, all worktape cells hold blanks, and every worktape head is positioned on the origin of its tape.

On a d-dimensional worktape, a box is a set of cells that form a d-dimensional cube. The volume of a box is the number of cells in it. The base cell of a box is the cell whose coordinates are the smallest. The relative position of a cell X in a box B is the list of coordinates of X when the base cell of B is taken as the origin.

Fix integers $d \geq 2$, $e \geq 2$, and h. Let E be an e-dimensional machine with h heads on one worktape. To establish Theorem 2, we design a d-dimensional simulator D with one head on a d-dimensional worktape and one head on a linear worktape. Technical details omitted from the presentation of the simulation in Section 3.1 appear in Sections 3.2 and 3.3.

3.1 Simulation

Consider a computation of N steps on E. As before, assume that N is available off-line. The coordinates of every cell accessed by E during the computation can be written as a binary string of length $O(\log N)$. For convenience write $\log^k N$ for $(\log N)^k$.

Define the function π by

$$\pi(x) = 2^{\lceil \log x \rceil} ;$$

if x is not a power of 2, then π maps x to the next larger power of 2. Set

$$b = \begin{cases} (N \log N)^{1/de} & \text{if } d \geq 3 \text{ or } e \geq 3 \\ (N/\log N)^{1/4} & \text{if } d = e = 2 \end{cases}$$

$$t = \pi((b \log N)^{1/d}),$$

$$u = \pi(b^{e/d}),$$

$$u^* = 4\, \pi((3^e h\, N \log N)^{1/d}) .$$

At the cost of introducing another constant factor, we may assume that b is an integer.

Partition the worktape of E into pairwise disjoint boxes of side b called blocks. The position of a block is the list of coordinates of its base cell.

Divide every cell of the d-dimensional worktape of D into three tracks numbered 0,1,2. In the initial configuration of D, the blank symbol in every cell is interpreted as a blank on each track.

A page is a box on the d-dimensional worktape whose side is a power of 2 between t and u that has a nonblank cell counter on track 0. We describe how the contents of a page P represent the contents of a simulated block B. (Abusing terminology, we shall say that P represents B when the configurations are implicitly specified.) First, the value of the nonblank cell counter must be at least the number of nonblank cells in B. To maintain the contents of these cells, P uses either a binary internal trie on track 1 or a literal representation on track 2.

An internal trie (sometimes called a digital trie [K3, p. 489]) is a collection of nodes organized into a binary tree according to the value of a binary key: the location of each node ν in the tree, written as a binary string $\beta(\nu)$ in a natural way as in Section 2, is an initial segment of the key. In the page P a node is a box of fixed volume $O(\log N)$ together with its contents. In the internal trie representation of B there is a node for every nonblank cell X in B, and the coordinates of cells are used as keys. The node ν for cell X holds on track 1

the coordinates of X (the key, written as a binary string),

the contents of X, and

the relative positions of the left and right children of ν (if any) in P.

The number of nodes equals the number of nonblank cells in B. (We may assume that E never writes a blank on a worktape cell.) See Figure 2. The internal trie representation is particularly useful when B has few nonblank cells.

The literal representation is straightforward. For every cell X of B there is a cell Y of P whose relative position in P is easily calculated from the coordinates of X such that Y holds the same symbol (on track 2) as X does. The literal representation is most efficient when B has many nonblank cells.

On the d-dimensional worktape D has a mass store, a directory, and a working area. The mass store is a box of side u* that holds the pages. The directory is a box of side u*/log N whose contents maintain the positions of pages in the mass store. During the simulation, D uses the position of a block B in E to retrieve from the directory the position of a page that represents B. (The directory is described in Section 3.2.) The working area comprises $3^e h$ contiguous boxes of side u at fixed positions; call these boxes slots. For $i = 1,\ldots,3^e h$, the position of the i-th slot can be calculated quickly. The coordinates of every cell in the mass store, the directory, and the working area can be specified by a string of length O(log N).

The linear worktape of D is used for routine arithmetic calculations, for counting, for maintaining the coordinates of the simulated heads of E, and for copying pages between the mass store and the working area.

Machine D simulates N steps of E in N/b phases of b steps each. At the beginning of the simulation of each phase, D determines the positions of the at most $3^e h$ blocks that surround the h simulated worktape heads of E; during the phase the heads remain within these blocks. For each of these blocks B_j, D calls procedure LOCATEPAGE, which uses the directory to locate the page P_j that represents B_j. (LOCATEPAGE is described in Section 3.2.) Next, D copies the contents of P_j into the j-th slot Q_j in the working area. In its finite-state control D maintains the correspondence between blocks and slots; on the linear worktape it records the positions of the corresponding pages P_j in the mass store. At the end of the phase, D copies the contents of pages from the working area to the mass store: D copies back to P_j just the box in Q_j that was written at the beginning of the phase. To send the head on the d-dimensional worktape to the proper locations, D uses the linear worktape as a counter.

During a phase, to simulate one step, D first decides for each simulated head H_i of E the block B_i in which H_i lies. Next, D recalls which slot Q_i in the working area corresponds to B_i. Using the contents of Q_i, machine D determines the contents of the cell X_i read by H_i and the new symbol that H_i writes on X_i. If Q_i represents B_i literally, then D calculates the relative position of the cell Y_i in Q_i that represents X_i, reads the contents of Y_i (on track 2), and writes the new symbol on track 2 of Y_i. If Q_i employs an internal trie to represent B_i, then D uses the coordinates of X_i to find the node that holds the contents of X_i on track 1 and records the new contents

of X_i in this node. If X_i has not been visited previously, then it holds a blank, and D creates a node for X_i at an appropriate location in the trie: D adds a node ν to the internal trie for which $\beta(\nu)$ is the shortest initial segment of the coordinates of X_i, written in binary, that is not the binary string $\beta(\nu')$ for a node ν' already in the trie.

Let us assess the time that D takes to simulate one step. First, D determines the coordinates of the worktape cells read by E (time $O(\log^2 N)$ for arithmetic calculations on the linear worktape because this worktape has just one head). If page Q in the working area uses a literal representation, then to retrieve or to write the contents of a cell, D computes the relative position of the representing cell in Q (time $O(\log^2 N)$) and moves its head across the working area (time $O(u)$) for a total of $O(u + \log^2 N)$ time. If Q uses an internal trie representation, then the head on the d-dimensional worktape visits $O(\log N)$ nodes of Q - at most one node for each bit of the coordinates of the cell. For each node, D copies its contents onto the linear worktape (time $O(\log N)$) and moves its head across the working area (time $O(u)$) for a total of $O((u + \log N)\log N)$ time. Thus, each phase takes time

$O(b(u + \log N))$ to simulate b steps

+ $O(\log^2 N)$ to calculate the positions of the $3^e h$ blocks that surround the heads of E

+ $O(u^*)$ to move the head across the mass store to each of $3^e h$ pages

+ $O(u^d)$ to copy the contents of $3^e h$ pages of volume u^d between the mass store and the working area

+ the time taken by calls to LOCATEPAGE.

In Section 3.2 we show that during the simulation, the total time consumed by calls to LOCATEPAGE is $O((N/b)u^*) + O(Nu \log N)$. Therefore, the

simulation runs in time

$$(N/b)[O(u^*) + O(u^d) + O(bu \log N)] + O(Nu \log N)$$
$$= O(Nu^*/b) \text{ by definition of } b$$
$$= O(N^{1+1/d-1/de} \log^{1/d-1/de} N) \text{ if } d \geq 3 \text{ or } e \geq 3$$
$$= O(N^{5/4} \log^{3/4} N) \text{ if } d = e = 2.$$

The simulation is routinely modified to handle head-to-head jumps by E.

Theorem 4. For all $d \geq 2$ and $e \geq 2$, every multihead e-dimensional machine with head-to-head jumps of time complexity t(n) can be simulated on-line by a machine with a d-dimensional worktape and a linear worktape, each with one access head, in time

$$O(t(n)^{1+1/d-1/de} \log^{1/d-1/de} t(n)) \text{ if } d \geq 3 \text{ or } e \geq 3,$$
$$O(t(n)^{5/4} \log^{3/4} t(n)) \text{ if } d = e = 2.$$

3.2 LOCATEPAGE and the Directory

The directory of D is a box of side $u^*/\log N$ on the d-dimensional worktape. To maintain the positions of pages whose contents represent contents of blocks, the contents of the directory are organized into an internal trie that uses the positions of blocks as keys. For every nonblank block B there is a node ν in the directory of volume O(log N) that holds

the position of B,

the position of the page P that represents B, and

the positions of the left and right children of ν (if any)

such that $\beta(\nu)$ is an initial segment of the position of B (written as a binary string of length O(log n)). We say that P is assigned to B. A page in the mass store is active if it is assigned to some block.

Throughout the simulation, several relationships among page sizes and nonblank cell counter values remain invariant. We assert these relationships as a lemma.

Lemma 2. During the simulation of E by D, the following always hold.

(a) If the nonblank cell counter of a page P has value m, then P has side $\min\{u, \pi((m \log N)^{1/d})\}$.

(b) Let page P represent block B. If P has side u, then P represents B literally. If the side of P is smaller than u, then P uses an internal trie.

(c) There are at most $3^e h\ N/b$ active pages.

(d) The sum of the nonblank cell counter values of the active pages is at most $3^e h\ N$.

Note that since $u^d \geq b^e$, if page P has side u, then P has at least as many cells as a block B and can contain a representing cell for every cell of B in a literal representation. If P has side $\pi((m \log N)^{1/d})$, where m is the value of its nonblank cell counter, then the at most m nodes of volume $O(\log N)$ in the internal trie representation, one for each nonblank cell of B, fit in P.

Furthermore, by Lemma 2(c), the directory can accommodate the nodes for the active pages. For each active page there is a node of volume $O(\log N)$. The implicit constant can be chosen so that

$$(3^e h\ N/b)O(\log N) \leq (u^*/\log N)^d.$$

Now we describe procedure LOCATEPAGE. Given the position of a block B, LOCATEPAGE employs the directory to find the page P in the mass store that represents B. If no page is assigned to B, then LOCATEPAGE calls procedure ALLOCATE to produce a blank box of side t in the mass store and sets the value of the nonblank cell counter of P to b; since B must be completely blank (it has not been previously visited), P already represents B.

If LOCATEPAGE found P in the mass store without calling ALLOCATE, then LOCATEPAGE ensures that P is large enough to include all nodes that D may add to the internal trie of P during the phase. Let p be the side of P and m be the value of its nonblank cell counter. LOCATEPAGE sets the value of the nonblank cell counter of P to $m' = \min\{m+b, b^e\}$. Let $p' = \min\{u, \pi((m' \log N)^{1/d})\}$. If $p < p'$, then the contents of P are copied into a new blank box of side p' (found in the mass store by ALLOCATE) to produce a larger page P' that represetns B with an internal trie. If $p < p' = u$, then the internal trie representation on track 1 of P' is converted into a literal representation on track 2. From every node of the internal trie, LOCATEPAGE extracts the coordinates of a cell X of B and copies the contents of X from track 1 into track 2 of the cell of P' that represents X in the literal representation. The other cells of B are blank, and track 2 of the corresponding cells of P' remains blank. Finally, LOCATEPAGE modifies the directory to assign P' to B. It is straightforward to confirm that the changes made by LOCATEPAGE satisfy the assertions of Lemma 2.

Let us evaluate the time taken by calls to LOCATEPAGE. Section 3.3 demonstrates that ALLOCATE operates in time $O(\log^2 N)$. In the directory, to move the head from one node to another takes time proportional to $u^*/\log N$, the side of the directory. Consequently, to determine the position of a page assigned to a block takes time

$$O(\log N)[O(\log N) + O(u^*/\log N)] = O(u^*)$$

because $O(\log N)$ nodes, each of volume $O(\log N)$ are accessed. The arithmetic calculations performed by LOCATEPAGE take time $O(\log^2 N)$.

We assess the time for converting internal trie representations into literal representations separately. Let page P represent block B, and suppose D converts the internal representation in P into a literal representation. Let m be the value of the nonblank cell counter of P when D performs the conversion. According to the definition of LOCATEPAGE, $m = \Theta(u^d/\log N)$. For each of at most m nonblank cells of B, machine D accesses $O(\log N)$ nodes of P in time $O((u + \log N)\log N)$. By Lemma 2(d), D converts at most $O(N/m) = O((N \log N)/u^d)$ pages during the simulation. Therefore, the total time for converting representations is

$$O((N \log N)/u^d)O(u^d/\log N)O((u + \log N)\log N) = O(Nu \log N).$$

We conclude that the total time consumed by calls to LOCATEPAGE during the simulation is

$$O(N/b)[O(u^*) + O(\log^2 N)] + O(Nu \log N) = O(Nu^*/b) + O(Nu \log N).$$

3.3 Storage Allocation

Machine D keeps a free storage list, a list of addresses of blank boxes in the mass store. For $q = 1,2,4,\ldots,u^*/2, u^*$, the free storage list has positions of at most 2^d-1 boxes of side q in the mass store. Initially, the free storage list holds the position of the mass store, a single box of side u^*.

Procedure ALLOCATE employs a buddy system [K1] to allocate a blank box with a desired side in the mass store. To obtain a blank box of side r, a power of 2, ALLOCATE finds the position of a box of side r on the free storage list. If the free storage list has no boxes of side r, then let q^* be the smallest power of 2 greater than r for which the free storage list has a box of side q^*. (We shall show that when ALLOCATE is called during the simulation, q^* must exist.) For $q = q^*,q^*/2,\ldots,4r,2r$ in order, select a position x_q of a box C_q of side q, delete x_q from the list, and add to the list positions of the 2^d pairwise disjoint boxes of side q/2 whose union is C_q. Finally, let x be the position of a box P of side r on the free storage list, delete x from the list, and return the value x. The time taken by ALLOCATE is $O(\log^2 N)$.

Lemma 3. Let r be a power of 2. Suppose the total volume of boxes on the free storage list is at least r^d in a configuration of D at the beginning of a call to ALLOCATE. Then this call can produce the position of a blank box of side r in the mass store.

Using Lemma 3, which can be proved routinely [L], we verify that ALLOCATE can always find the requested boxes in the mass store. At any point during the simulation, let $P_1, P_2, \ldots$ be the active pages in the mass store and $m_1, m_2, \ldots$ be the values of their nonblank cell counters. Let P_j be assigned to block B_j. According to Lemma 2(a), the side of P_j is at most $\pi((m_j \log N)^{1/d})$. The mass store holds smaller inactive pages that were assigned to B_j in previous configurations of D. Because the volumes of these smaller pages are distinct powers of 2, the total volume of pages ever assigned to B_j is at most twice the volume of P_j, hence at most $2(\pi((m_j \log N)^{1/d}))^d$. By Lemma 2(c), the total volume of all pages in the mass store is at most

$$\Sigma_j 2(\pi((m_j \log N)^{1/d}))^d \leq 2(2^d 3^e h\ N \log N) \leq (u^*)^d .$$

Ergo, whenever ALLOCATE is invoked to find a blank box of side r, the total volume of blank boxes on the free storage list is at least r^d.

4. RESEARCH DIRECTIONS

1. Kosaraju [Ko] proved that multihead one-dimensional machines with head-to-head jumps can be simulated in real time by one-dimensional machines with just one access head per worktape. Do multihead tree machines and multidimensional machines with head-to-head jumps share this property? Leong and Seiferas [LS] presented real time simulations of multihead multidimensional machines without head-to-head jumps.

2. How fast can tree machines and multidimensional machines simulate each other? Reischuk [R] devised on-line simulations of multidimensional machines of time complexity $t(n)$ by tree machines in time $O(t(n)c^{\log^* t(n)})$ for a constant c. Pippenger and Fischer [PF] gave an on-line simulation of a tree machine by a one-dimensional machine in time $O(t(n)^2/\log t(n))$. For $d \geq 2$, a theorem of Grigoriev [G] implies that every tree machine can be simulated on-line by a d-dimensional machine in time $O(t(n)^{1+1/(d-1)})$, but no better upper bound is known.

REFERENCES

[G] D. Ju. Grigor'ev. Imbedding theorems for Turing machines of different dimensions and Kolmogorov's algorithms. Soviet Math. Dokl. 18 (1977) 588-592.

[HS] F. C. Hennie and R. E. Stearns. Two-tape simulation of multitape Turing machines. J. ACM 13 (1966) 533-546.

[K1] D. E. Knuth. The Art of Computer Programming, vol. 1: Fundamental Algorithms. Addison-Wesley, 1968.

[K3] D. E. Knuth. The Art of Computer Programming, vol. 3: Sorting and Searching. Addison-Wesley, 1973.

[Ko] S. R. Kosaraju. Real-time simulation of concatenable double-ended queues by double-ended queues. Proc. 11th Ann. ACM Symp. on Theory of Computing, 1979, pp. 346-351.

[LS] B. L. Leong and J. I. Seiferas. New real-time simulations of multi-head tape units. J. ACM 28 (1981) 166-180.

[L] M. C. Loui. Simulations among multidimensional Turing machines. Tech. Rep. TR-242, Lab. for Comp. Sci., M. I. T., Aug. 1980.

[PSS] W. J. Paul, J. I. Seiferas, and J. Simon. An information-theoretic approach to time bounds for on-line computation. To appear in J. Comp. Sys. Sci.

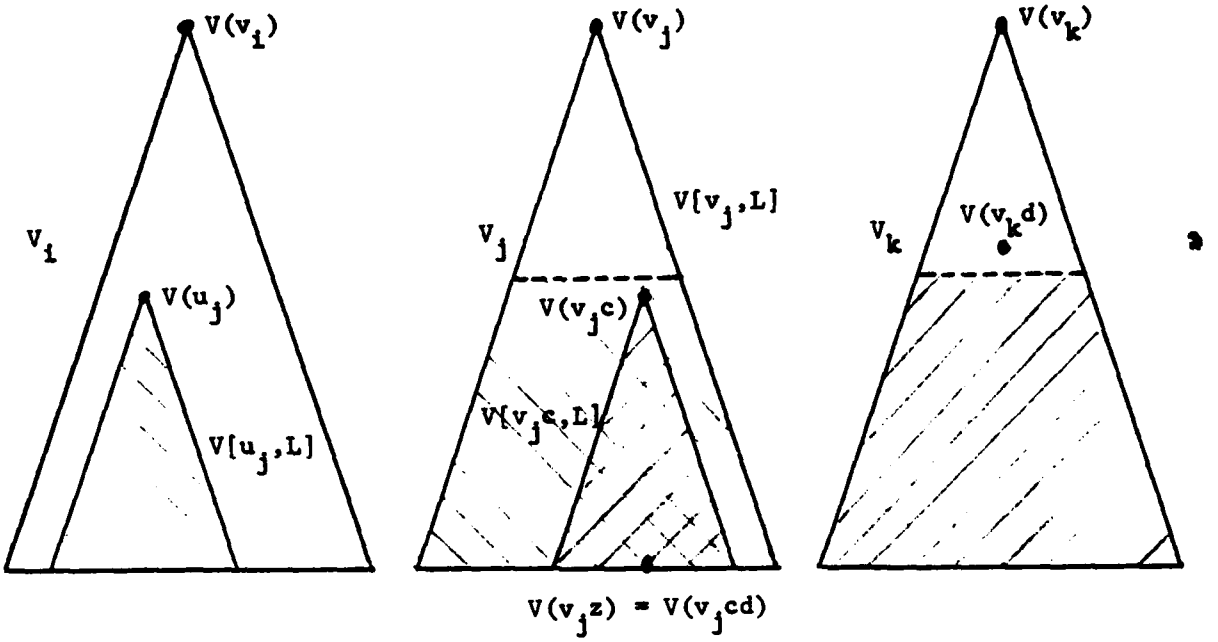
[PR] W. J. Paul and R. Reischuk. On time versus space II. Proc. 20th Ann. Symp. on Foundations of Computer Science, 1979, pp. 298-306.

[PF] N. Pippenger and M. J. Fischer. Relations among complexity measures. J. ACM 26 (1979) 361-381.

[R] R. Reischuk. A fast implementation of a multidimensional storage into a tree storage. Proc. 7th Intern. Colloq. on Automata, Languages, and Programming, Springer-Verlag, 1980, pp. 531-542.

[S] H.-J. Stoss. Zwei-Band Simulation von Turing Maschinen. Computing 7 (1971) 222-235.

Figure 1.


V(v_i)
V_i
V(u_j)
V[u_j,L]
V(v_j)
V_j
V[v_j,L]
V(v_jc)
V(v_jc,L]
V(v_k)
V_k
V(v_kd)
V(v_jz) = V(v_jcd)

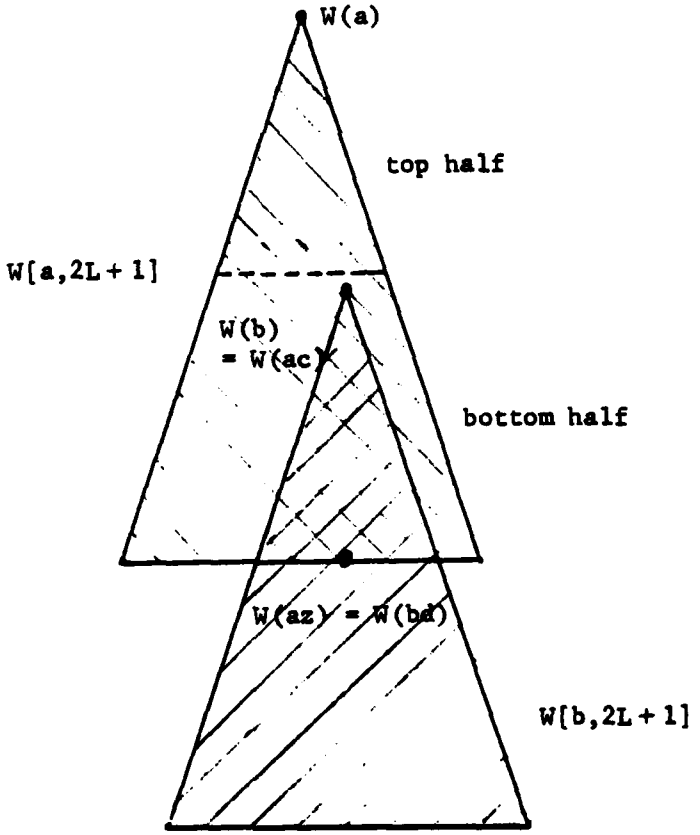
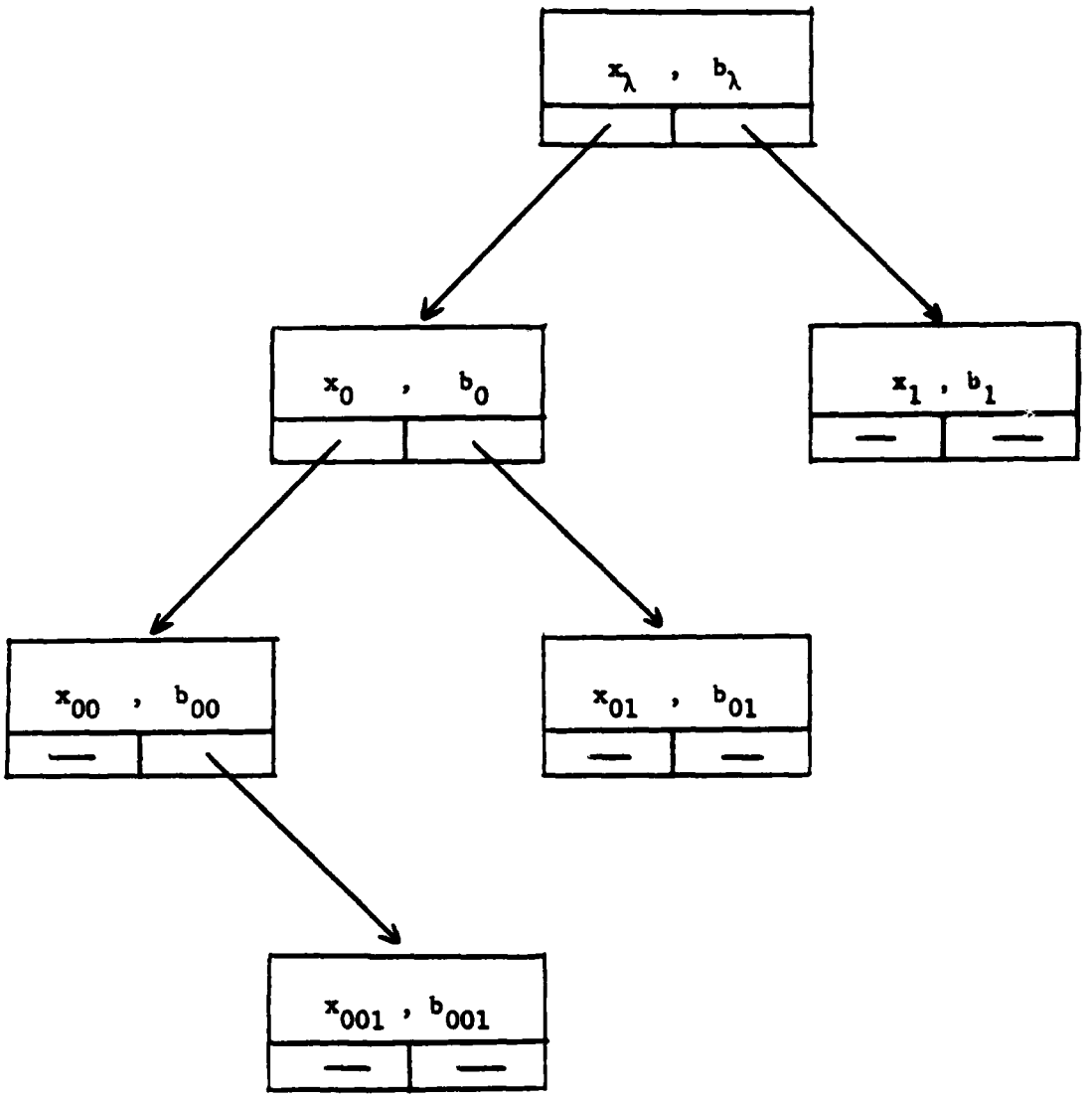
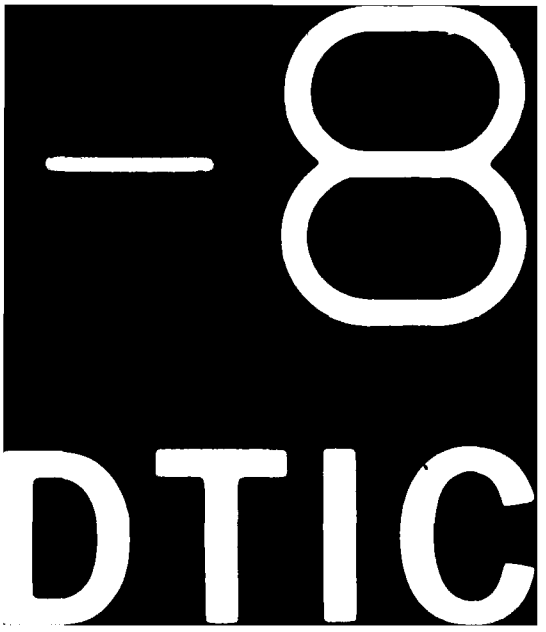

W(a)
top half
W[a,2L+1]
W(b)
= W(ac)
bottom half
W(az) = W(bd)
W[b,2L+1]

Figure 2. An internal trie representation. The cell at coordinates x_{001} of E contains the symbol b_{001}. The string 001 is an initial segment of x_{001} (written in binary). λ denotes the empty string.




8
DTIC